LITERATURE REVIEW:
Explore how the OpenMP can improve application performance

Dehui Yu
School of Computer Science
University of Ottawa
Ottawa, Canada K1N 6N5
dyu105@uottawa.ca

October 7, 2022

1 Introduction

Parallel computing has been around for many years, but recently interest in the high performance computing community has begun to grow. This is because hardware is getting cheaper and cheaper.In my project, I want to explore the use of OpenMP.OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared-memory multiprocessing programming in C, C++, and Fortran,on many platforms, instruction-set architectures and operating systems, including Solaris, AIX, FreeBSD, HP-UX, Linux, macOS, and Windows.

My project wants to use Monte Carlo algorithms to determine the effectiveness of OpenMP in running speed. What is Monte Carlo method? When the problem to be solved is the probability of a certain random event or the expected value of a certain random variable, the probability of this random event is estimated by the frequency of this event through some "experimental" method, or some numerical characteristics of this random variable are obtained and taken as the solution of the problem.Usually, Monte Carlo methods solve various mathematical problems by constructing random numbers that conform to certain rules. The Monte Carlo method is an effective way to find numerical solutions for problems that are too complex to obtain analytical solutions or for which no analytical solutions are available at all. The most common application of the general Monte Carlo method in mathematics is the Monte Carlo integral.

Case 1: Calcualte PI using Monte Carlo algorithms.
The more the number of points scattered, the more accurate the PI will be. However, there will be a problem at this time, that is, after too many points, the algorithm will take a long time to run. And exactly the simulation of the scatter point this thing can be implemented in parallel, that is to say, the scatter point does not affect, so you can let multiple processes or threads scatter points at the same time, and finally all into/thread scatter points together statistics, to get a higher accuracy of PI.

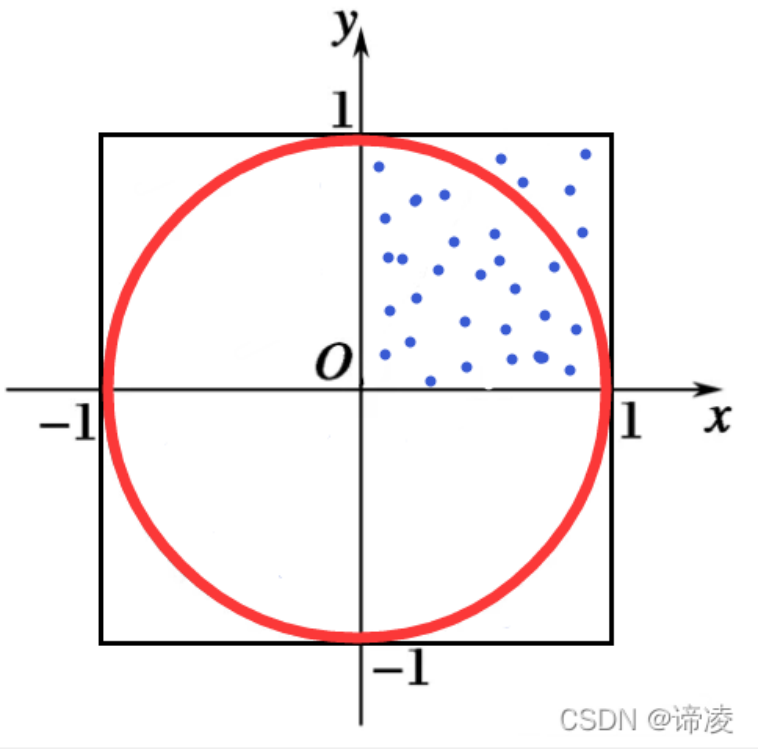

Figure 1: Monte Carlo Algorithm

2 Literature Review

After I came up with this idea, my first step was to investigate why and how to use OpenMP. This article[1] tells me why we should run our code in parallel, and the answer is obvious: to reduce the time it takes to run. This article introduces an interesting point that simply adding more processing won't significantly improve the runtime of your code. Figure 2 shows the scalability of parallelised codes. It clearly shows that simply adding more processors will not always make a significant improvement to the run time of a code.Therefore, in our project, we will also explore the optimal Scalability of Parallelisation. At the same time, it showed me that OpenMP works in C, C++, and Fortran, which allowed me to confirm the project language.

This article[4] made me want to use OpenMP to make this algorithm more efficien,To solve a problem, first a probabilistic model or stochastic process is built such that its parameters or numerical characteristics are equal to the solution of the problem: then these parameters or numerical characteristics are calculated by observing the model or process or sampling experiments, and finally an approximation of the solution is given. The accuracy of the solution is expressed as the standard error of the estimate.

Before reading this paper[3], I didn't know the difference between OpenMP and MPI. OpenMP and MPI are two approaches to parallel programming, and the following figure 3 illustrates the difference. As can be seen from the figure, MPI is designed for distributed,

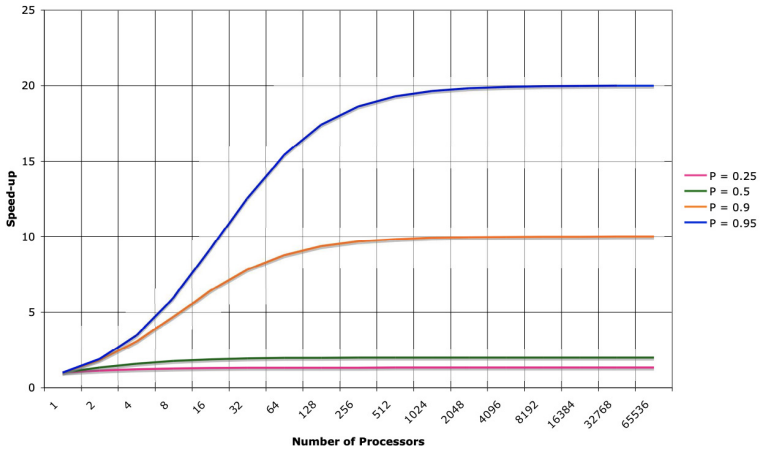

Figure 2: Scalability of parallelised codes

while OpenMP is designed for shared memory. Since it is two parallel computing methods, this leads me to also compare OpenMP and MPI methods in my paper, which one is better?

This article[2] is also a very interesting article that I have read. It invented An OpenMP Tools Application Programming Interface for Performance Analysis. Before it was introduced, OpenMP did not provide efficient performance tools. OMPT provides three main functions: (1) the runtime state tracking, it makes the performance based on the sampling tools to know what the application thread is doing, (2) the callbacks and query function, based on the performance of the sampling tools application performance can not be attributed to complete the invocation context, (3) additional correction notice, you can build a fully functional monitoring function. It is the fervent hope of the Tools Working Group that OMPT is adopted as part of the OpenMP standard and supported by all compliant OpenMP implementations. I'll consider using it as our analysis tool in our tests of Monte Carlo algorithms.

In fact, before the above official tools were proposed, there were some researchers who could not develop their own performance analysis tools: ompP, POMP, etc. This article[5] investigates Performance Tools for OpenMP. This article discusses the pros and cons of these performance profiling tools in great detail, and I may use one or more of these tools for performance profiling on my projects.

OpenMP has a lot of practical uses, which makes the computing speed a lot faster. Such as: Paper[7] Disposing an Optimal Strategy to Parallelize the Solution to Large 3D Magneto-quasi-static (MQS) Problems by combining the MPI and OpenMP approaches.Article[6] presents a novel hybrid graph coloring algorithm and discuss how to obtain the best performance on such systems from algorithmic, system and engineering perspectives by using OpenMP. For the most part, these papers show a speed boost when using OpenMP, but many of them don't talk about what happens when you increase the number of parallel computing nodes infinitely.This is one of the highlights of my project because I will take care of it.

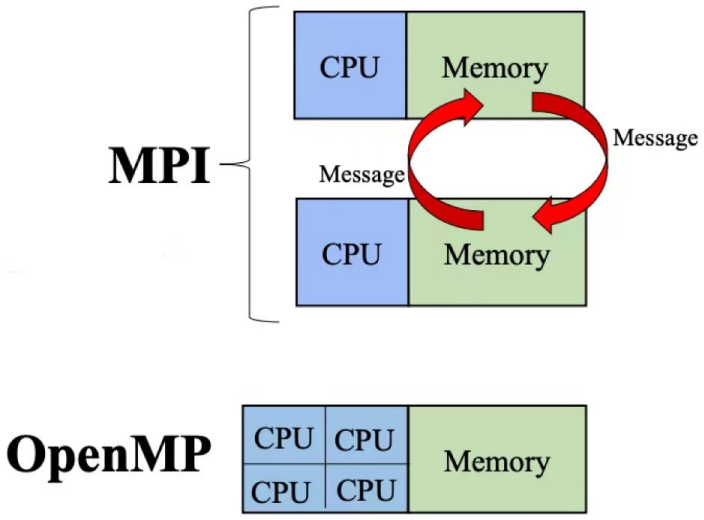

Figure 3: OpenMP vs MPI

3 Summary

This is the Literature review I wrote after reading some articles. I believe there will be many unexpected problems and indicators in the actual calculation. To sum up, my project is to explore the application of OpenMP in Monte Carlo algorithms and compare the performance of OpenMP and MPI methods. Besides that, I will also explore the change in speed after infinitely increasing points, and I believe my idea is a good one. Thanks for reading my literature review.

References

[1] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[2] Alexandre E Eichenberger, John Mellor-Crummey, Martin Schulz, Michael Wong, Nawal Copty, Robert Dietrich, Xu Liu, Eugene Loh, and Daniel Lorenz. Ompt: An openmp tools application programming interface for performance analysis. In *International Workshop on OpenMP*, pages 171–185. Springer, 2013.

[3] Haoqiang Jin, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman. High performance computing using mpi and openmp on multi-core parallel systems. *Parallel Computing*, 37(9):562–575, 2011.

[4] Werner Krauth. Introduction to monte carlo algorithms. In *Advances in Computer Simulation*, pages 1–35. Springer, 1998.

[5] Mubrak S Mohsen, Rosni Abdullah, and Yong M Teo. A survey on performance tools for openmp. *World Academy of Science, Engineering and Technology*, 49:754–765, 2009.

[6] Ahmet Erdem Sariyüce, Erik Saule, and Ümit V Çatalyürek. Scalable hybrid implementation of graph coloring using mpi and openmp. In *2012 IEEE 26th International*

Parallel and Distributed Processing Symposium Workshops & PhD Forum, pages 1744–1753. IEEE, 2012.

[7] Salvatore Ventre, Francesca Cau, Andrea Chiariello, Gaspare Giovinco, Antonio Maffucci, and Fabio Villone. Fast and accurate solution of integral formulations of large mqs problems based on hybrid openmp–mpi parallelization. *Applied Sciences*, 12(2):627, 2022.